INSTRUCTIONS :

1. Complete all questions in your designated project group.
2. All members must contribute to writing the codes. (i.e. 1 question = 1 person, and share the workload if there's an additional question relative to the actual number of members in your team (i.e. 5)). Ensure that all members must understand and explain codes from any of the questions.
3. During viva, all students in each team will be randomly asked to describe, answer and edit any of the answers provided. Marks will be given to your ability to present the answers.

Lab Report

Prepare a report to solve the above problems. The report should contain all the sections as below for each question:

Section	Description
1. Problem	Description on the problem
2. Solution	Explanation on how to solve the problems below
3. Sample Input & Output	A few sets of input and output (snapshot)
4. Source code	Java source code

Requirements

1. Group Assignment (Grouping is the same as your project group)
2. Cover page that includes all student matric number and full name.
3. Font: Times New Roman 12, Line Spacing: 1 ½ Spacing
4. Submit to Spectrum according to your OCC. **Deadline** : Before your viva session (W9).

Question 1: Nearest Prime Number Finder

Problem Statement:

You are fascinated by prime numbers and often wonder which prime is closest whenever you see a random number. For example, if you see 21, you might ask yourself whether 19 or 23 is nearer. To explore this curiosity, you decide to write a Java program that allows a user to enter any integer and automatically finds the nearest prime number. If the entered number is already prime, the program will return it. In cases where two primes are equally close, it will display the smaller one. This program will help you quickly identify the nearest prime for any number you encounter.

You must implement and use the following methods:

1. boolean isPrime(int num)
2. int getNearestPrime(int num)

*****You may **create and use any additional helper methods** if necessary.*

Input:
Enter any integer number.

Output:
Display the nearest prime number to the given input. If the number itself is prime, display the number itself.

Sample Output:

Input	**Output**
21	19
153	151
11	11

Question 2: Message Encoder

Problem Statement:

It was computer class at Hogwards Secondary School, and students were busy practicing typing and basic Excel skills. But at the back of the lab, Ryan and Ali were secretly chatting through a shared file. After nearly being caught by their teacher, Mr. Tan, they decided to create a secret message encoder to hide their chats if he checked their screen.

During recess, they came up with a clever three-step plan for their encoder. First, after every character in the string, they would insert a random special character like @, #, or $ to confuse anyone reading it. Next, they would split the string into two halves, reverse each half, and rejoin them — keeping the middle character unchanged if the length was odd. Finally, they would shift each character's ASCII value depending on its type: uppercase letters by (2 + index), lowercase letters by (3 + index), digits by (10 + index), and leave symbols unchanged.

As Ryan, your task is to write a Java program that performs this encoding process to transform any given message into a secret message.

The next day, when Mr. Tan walked by, Ryan quickly ran their encoder. Their chat instantly turned into a line of strange symbols and numbers. Mr. Tan glanced at their screen, nodded, and moved on — completely unaware that a secret conversation had just been hidden in plain sight.

Hint:
You can refer to online ASCII converters or random symbol generators to test your logic for symbol insertion and character shifting.

You must implement and use the following methods:

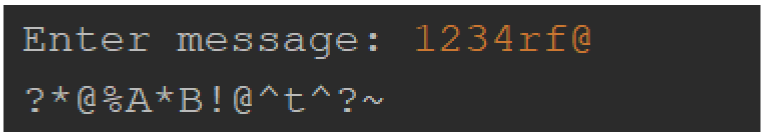
1. String insert(String str)
2. String breakAndFlip(String str)
3. String shift(String str)

***You may *create and use any additional helper methods* if necessary.**

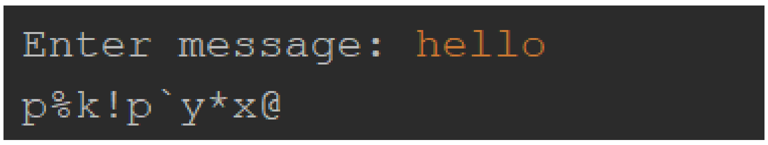
Input:
Enter a message string to encode.

Output:
Display the encoded secret message after performing all three steps.

Sample Output:



```
Enter message: 1234rf@
?*@%A*B!@^t^?~
```



```
Enter message: hello
p%k!p`y*x@
```

Question 3: Mean Adjustment

Problem Statement:

In a biology laboratory, you are helping Professor Lim analyze the results of a bacterial growth experiment. After recording several sample measurements, the average growth rate does not match the expected target mean. To investigate possible experimental errors, Professor Lim asks you to write a program that adjusts the dataset by removing certain samples so that the new mean becomes as close as possible to the target mean.

Your program should allow the user to enter the total number of samples, the maximum number of samples that can be removed, the target mean, and the list of sample values. It must validate that the maximum removable samples are less than the total number of samples. Then, the program should repeatedly remove one sample at a time, up to the specified limit that brings the mean closest to the target mean. After each removal, display which sample was removed and the new mean. Finally, show the final mean and its difference from the target mean.

This simulation helps the research team identify which data points most affect the experiment's results and understand how removing certain samples changes the overall average.

You must implement and use the following methods:

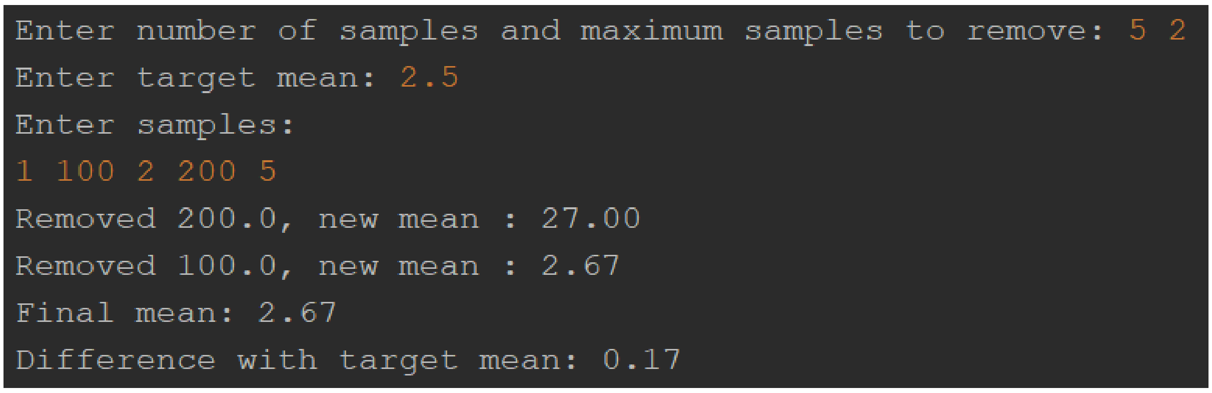
1. double getMean(double[] numbers)
2. int getBestRemovalIndex(double[] numbers, double targetMean)
3. double getDifference(double[] numbers, double targetMean)

******You may* ***create and use any additional helper methods*** *if necessary.*

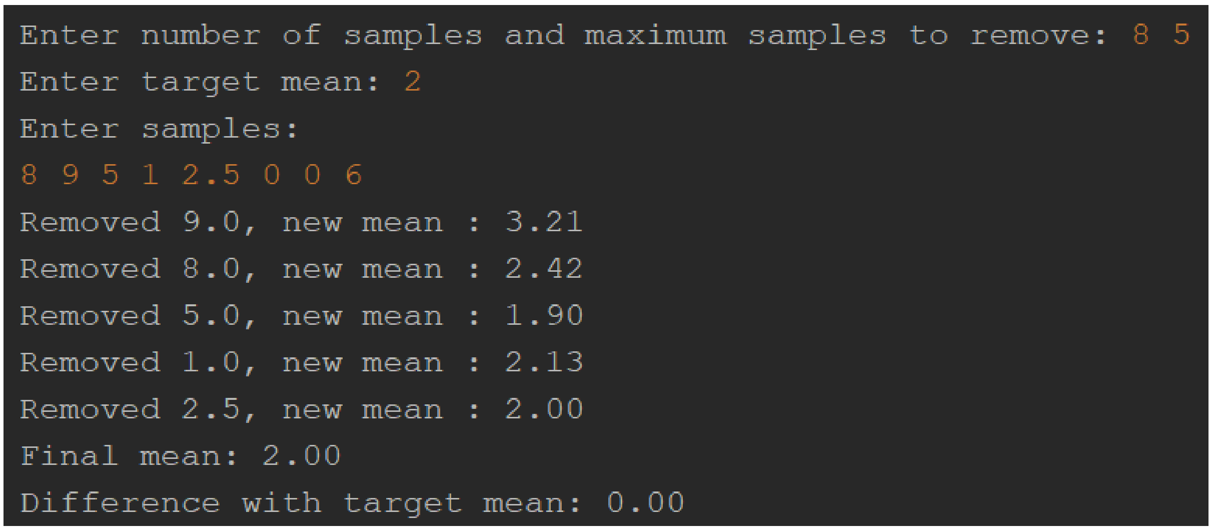
Input:
Enter the total number of samples, the maximum removable samples, the target mean, and the list of sample values.

Output:
Display the sample removed at each step, the updated mean after each removal, the final mean, and its difference from the target mean.

Sample Output:



```
Enter number of samples and maximum samples to remove: 5 2
Enter target mean: 2.5
Enter samples:
1 100 2 200 5
Removed 200.0, new mean : 27.00
Removed 100.0, new mean : 2.67
Final mean: 2.67
Difference with target mean: 0.17
```



```
Enter number of samples and maximum samples to remove: 8 5
Enter target mean: 2
Enter samples:
8 9 5 1 2.5 0 0 6
Removed 9.0, new mean : 3.21
Removed 8.0, new mean : 2.42
Removed 5.0, new mean : 1.90
Removed 1.0, new mean : 2.13
Removed 2.5, new mean : 2.00
Final mean: 2.00
Difference with target mean: 0.00
```

Question 4: Triangle Calculation

Problem Statement:

During your mathematics class, your teacher gives you a challenge: “Can you create a program that helps students check their triangle calculations?” You decide to take it up and write a Java program that determines the basic information of a triangle based on user input.

Your program can work in two ways:

1. SSS (Side-Side-Side): When all three sides are known.
2. SAS (Side-Angle-Side): When two sides and the included angle are known.

Using the provided values, your program should first validate whether a triangle can exist (based on the rule that the sum of any two sides must be greater than the third). If it’s valid, your program then applies the Law of Cosines to find any missing sides or angles, and uses Heron’s Formula to calculate the triangle’s area.

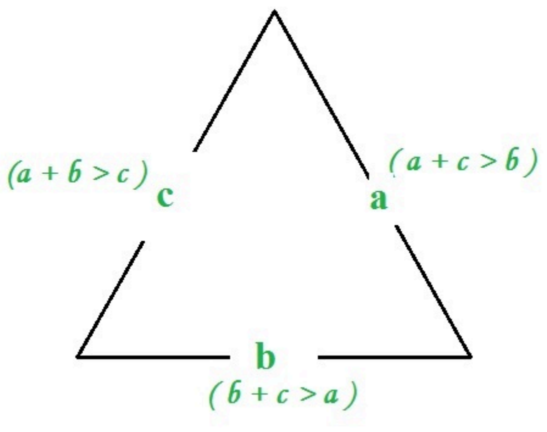
Finally, the program displays all the sides, angles, and the area of the triangle. If the values do not form a valid triangle, it simply shows a message: “Not a valid triangle.”

Related Formulas:

1. Law of Cosines (for Sides and Angles)
Used to find a missing side or angle in a triangle.

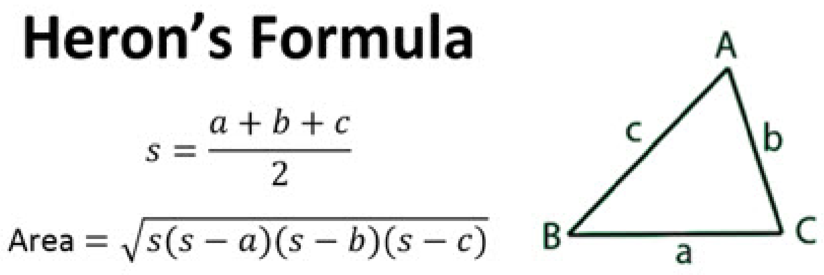
To find a missing side (SAS):	To find a missing angle (SSS):
$a^2 = b^2 + c^2 - 2bc\cos A$	$\cos A = \dfrac{b^2 + c^2 - a^2}{2bc}$
$b^2 = a^2 + c^2 - 2ac\cos B$	$\cos B = \dfrac{a^2 + c^2 - b^2}{2ac}$
$c^2 = a^2 + b^2 - 2ab\cos C$	$\cos C = \dfrac{a^2 + b^2 - c}{2ab}$

2. Triangle Validity Rule
A triangle is valid if and only if the sum of any two sides is greater than the third side:



3. Heron's Formula (for Area)

Used when all three sides are known.



Heron's Formula

$$s = \frac{a+b+c}{2}$$

$$\text{Area} = \sqrt{s(s-a)(s-b)(s-c)}$$

You must implement and use the following methods:

1. double getSideBySAS(double side1, double angle, double side2)
2. double getAngleBySSS(double side1, double side2, double targetSide)
3. double getArea(double a, double b, double c)

**You may create and use any additional helper methods if necessary.*
You may use the **Java Math library (e.g., Math.pow(), Math.sqrt(), Math.cos(), Math.toRadians(), etc.) wherever necessary to perform mathematical calculations.*

Input:

Enter the triangle type (SSS or SAS) followed by the required values (sides and/or angle).

Output:

Display all sides, angles, and the area of the triangle rounded to 2 decimal places if the triangle is valid. If invalid, print "Not a valid triangle."

Sample Output:

Input	**Output**
(SSS) Side 1: 9 Side 2: 25 Side 3: 17	Triangle Info Side A: 9.00 Side B: 25.00 Side C: 17.00 Angle BC: 11.48 Angle AC: 146.44 Angle AB: 22.08 Area of Triangle: 42.29
(SSS) Side 1: 8 Side 2: 17 Side 3: 9	Triangle Info Side A: 8.00 Side B: 17.00 Side C: 9.00 Not a Valid Triangle
(SAS) Side 1: 6 Angle: 60 Side 2: 6	Triangle Info Side A: 6.00 Side B: 6.00 Side C: 6.00 Angle BC: 60.00 Angle AC: 60.00 Angle AB: 60.00 Area of Triangle: 15.59

Question 5: Number Base Conversion

Problem Statement:

Your little brother has just started learning about number bases in his secondary school math class. He often comes to you, frustrated, trying to check if his answers for base conversions are correct, especially when the numbers involve letters or fractions.

To make things easier for him (and save yourself from constant checking), you decide to write a Java program that can handle any kind of number conversion — from base 2 all the way up to base 36. The program can take whole, fractional, and even negative numbers, and convert them accurately into another base. You also make sure it can round fractional parts up to 8 digits after the decimal point.

The program first asks for the number and its base, then validates that all digits are valid for that base. After that, it prompts for the new base and displays the result, instantly showing your brother whether his answer is right or wrong.

You must implement and use the following methods:

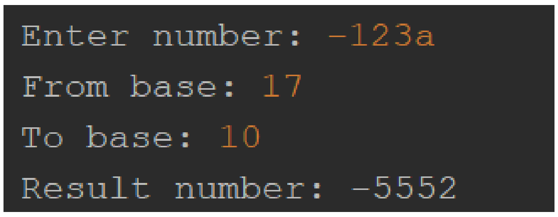
1. double convertToBaseTen(String num, int base)
2. String convertFromBaseTen(double num, int base)

You may **create and use any additional helper methods if necessary.*

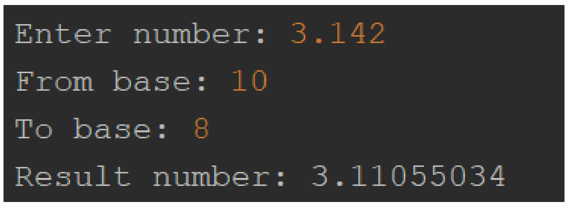
Input:
Enter the number, its current base, and the target base for conversion.

Output:
Display the number converted from the original base to the target base, rounded to 8 decimal places if it contains a fractional part.

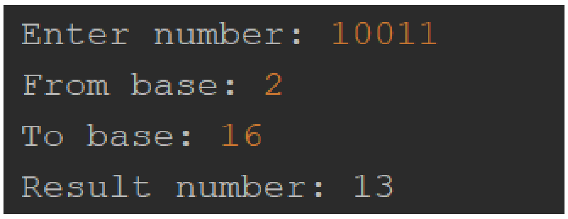
Sample Output:



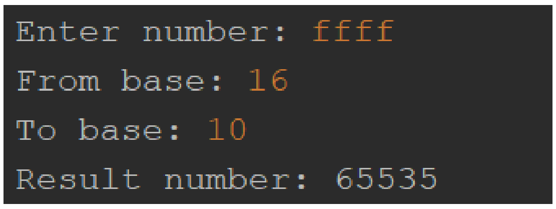
```
Enter number: -123a
From base: 17
To base: 10
Result number: -5552
```



```
Enter number: 3.142
From base: 10
To base: 8
Result number: 3.11055034
```



```
Enter number: 10011
From base: 2
To base: 16
Result number: 13
```



```
Enter number: ffff
From base: 16
To base: 10
Result number: 65535
```

Question 6: Palindrome Generator

Problem Statement:

You are required to write a Java program that determines whether a given string can be rearranged to form a palindrome, and displays all possible palindrome arrangements.

A palindrome is a word that reads the same forwards and backwards such as level, madam and noon.

Your program should:

1. Prompt the user to enter a string (treat all uppercase letters as lowercase letters).
2. Check whether the string can be rearranged into a palindrome.
3. If it cannot form a palindrome, display "Cannot be palindrome."
4. If it can form a palindrome, display all possible palindrome arrangements.

You must implement and use the following methods:

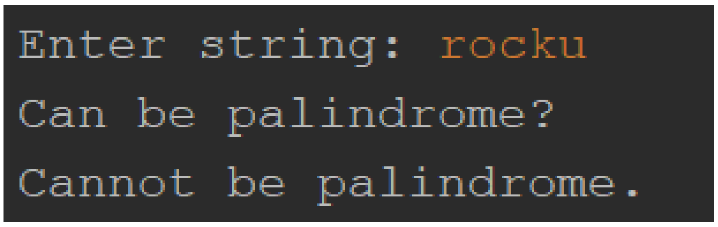
1. boolean canbePalindrome(String str)
2. void getlPossiblePalindrome(String str)

You may **create and use any additional helper methods if necessary.*
You are **not allowed to use ArrayList, HashSet, or any other advanced Java collection classes.*

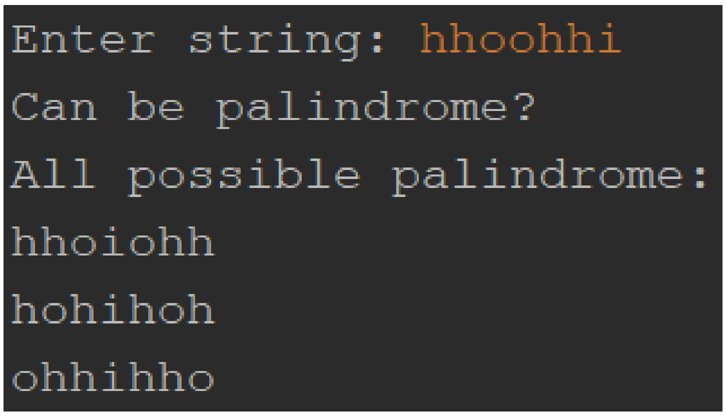
Input:
Enter a string to check if it can be rearranged into a palindrome. All uppercase letters will be treated as lowercase.

Output:
Display "Cannot be palindrome." if the string cannot be rearranged into a palindrome. Display all possible palindrome arrangements if it can be rearranged into a palindrome.

Sample Output:



```
Enter string: rocku
Can be palindrome?
Cannot be palindrome.
```



```
Enter string: hhoohhi
Can be palindrome?
All possible palindrome:
hhoiohh
hohihoh
ohhihho
```